Sponsored by the Defense Advanced Research Projects Agency (DoD)
DARPA/CSTO

Title of Contract:
"A Proposed Research Program in Strategic Computing"

Project:
SIMS: Single Interface to Multiple Systems

DARPA Order No. 6096
Issued by DSS-W under Contract
MDA903-87-C-0641

Period of Performance: 09/01/87 – 08/31/92

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies either expressed or implied of the Defense Advanced Research Projects Agency or the U.S. Government.

1 Task Objectives

The ultimate goal of the SIMS project was to relieve a user of computer services of the need to be familiar with the various services available. In the SIMS view, the user should make requests as though he or she were dealing with a single, seamless service. SIMS analyzes the user request, determines the necessary calls on the various services, and executes them. As a consequence, SIMS must devote considerable effort to the planning mechanism that establishes the sequence of service calls that will satisfy the user's request.

Most tasks performed by users of information systems involve interaction with multiple software servers. Examples can be found in the areas of resource planning and briefing applications, in analysis (both of intelligence data and logistics forecasting), and in command and control. A typical task of this kind may involve retrieval of data from several databases, data interpretation, numerical calculations, and report generation. It may also involve incorporation of information generated by expert systems. The result of processing data from one source may have to be fed as input to another.

Besides performing different functions, servers will usually also differ considerably in their input languages. The user of multiple services must be familiar with them all in order to decompose his/her task into commands to the various servers and then in order to provide the required input to each server. We call this problem the *software integration* problem.

Until now, the only approach to the software integration problem has been to build custom systems that support an integrated view of the underlying servers. Such systems are expensive, are suitable only for the application for which they were crafted, are relatively inflexible with respect to functionality, and are difficult to modify. For example, the incorporation of an additional server may very well require rewriting the entire custom integration system, with no obvious way to take advantage of the experience gained when writing the existing version. In contrast, SIMS would allow maintainers to assimilate additional servers into the system through a much less painful process of building a declarative model of that server, allowing the SIMS planner to perform the integration task.

The SIMS system has been applied to the domain of information needed for daily Naval briefings to the Commander in Chief of the Pacific Fleet. Three (simulated) databases, containing information about ships' locations, activities, and readiness status, are accessed by the system.

2 Technical Approach

SIMS attempts to solve the problem of software integration by dispensing with the custom system approach and providing instead for *logical integration.*[1] Within SIMS, the various

[1] An earlier version of SIMS, of which the current one is an extension and an enhancement, is described in [Pavlin88a, Pavlin88b].

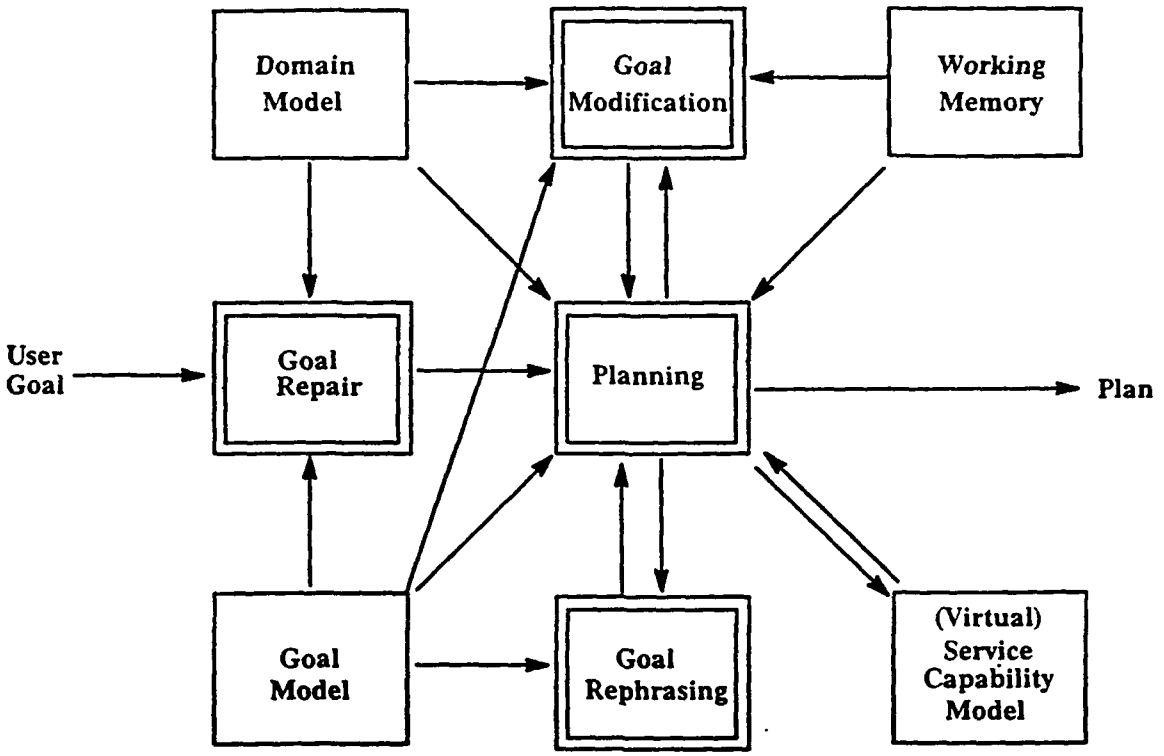

Figure 1: SIMS Overview Diagram

services remain separate. In order to make use of them, their capabilities must be represented in some common way. Thus, their capabilities are modeled in the form of *plan operators*. Each operator represents a type of request, or command, that may be issued to one of the servers. In addition, SIMS models the application domain and possible user goals, for reasons that will be explained below.

Figure 1 describes the flow information among the various processing and model components of SIMS. SIMS' components are described in more detail later in the paper.

The user presents a goal to the system in a *server-independent interaction language*. The user's goal is first inspected to ascertain that its form is acceptable. Since SIMS insulates the user from the available services, it is expected that user goals will often not be well formed: they might refer to objects that are not represented in the model, or might use wrong names for objects that are. Goals that are not well-formed will have to undergo *Repair*, one of three types of *goal reformulation*[2] that SIMS supports. Information about goals is represented explicitly within SIMS' model, and is used to support goal repair.

Once a correct form for the goal is established, it activates SIMS' planner, which attempts

[2]In a slightly different context, this idea of goal reformulation was raised in [Neches85].

to achieve it using the available operators. If a plan is successfully constructed, SIMS then performs it, accessing databases and programs, integrating the results, and presenting them to the user.

If the planner fails to devise a plan for a given goal, SIMS will engages in further processes of goal reformulation. When a goal is not immediately tractable, it may be possible to analyze relations between that goal and others known to SIMS, and to come up with a different goal (or collection of goals) whose achievement would subsume the original one. This process is called *Goal Rephrasing.* If the user's goal can be reformulated, an additional attempt is made to plan for the new goal(s). If this is still unsuccessful, SIMS may attempt to *Modify* the goal, replacing it with a different goal that achieves a similar purpose. This type of goal reformulation is not currently implemented.

The plan finally obtained can then be executed, and the results presented to the user.

In [Swartout88], Drew McDermott notes that

> Every planning method we know of relies on dividing a problem into pieces, producing plans for the pieces, and reconciling them. It is hard to divide an arbitrary statement (replete with quantifiers, connectives, and modalities) into pieces that make sense.

SIMS' explicit representation of goals, and relationships they may have to each other, goes a long way towards allowing a planner to succeed even when it can find no way to divide a given goal. In such cases, the SIMS planner may be able to use knowledge available to it about the semantics of goals to replace one with a more tractable other.

The following section describes SIMS' planning, modeling, and the goal reformulation processes in more detail.

3 General Methodology

Develop the theoretical approach hand-in-hand with a working implementation of the theory in a computer program.

4 Technical Results

4.1 Summary of Results

SIMS has made progress along all significant dimensions of its approach to the software integration problem:

- SIMS operates using a general purpose planner, for which individual queries to databases serve as operators.
- All data services available through SIMS are accessed uniformly by stating requests in the LOOM knowledge representation language. Requests are analyzed using both built-in LOOM reasoning facilities and SIMS-specific ones.
- SIMS has developed models of four different database tables containing actual (sanitized) information from the CINCPACFLT domain. The data in these tables is accessed and combined without and modification to the DBMS itself.
- Several goal reformulation strategies have been developed and implemented. In some cases, SIMS can correct syntactically flawed user requests. It is capable of replacing goals which it cannot achieve with equivalent combinations of goals which it can.

In order to perform reformulations and to plan the sequence of server calls that will achieve a given user's task, SIMS' planner must access the representations of the application domain, of user goals, and of server capabilities. This information is described using the general-purpose knowledge representation language which is described in the next section.

4.2 Model Implementation: Loom

Loom[MacGregor88] is a language and environment for constructing intelligent applications. It combines features of both frame-based and semantic network languages, and provides some reasoning facilities. Loom is used for all modeling and most reasoning in SIMS.

The heart of the Loom system is a powerful knowledge representation system, which is used to provide deductive support for the declarative portion of the Loom language. Declarative knowledge in Loom consists of definitions, rules, facts, and default rules. A deductive engine called a *classifier* utilizes forward-chaining, semantic unification and object-oriented truth maintenance technologies in order to compile the declarative knowledge into a network designed to efficiently support on-line deductive query processing.

The Loom system includes a semantic pattern matcher for interpreting production rules, and a pattern-directed method dispatching facility that supports the definition of object-oriented methods. The high degree of integration between Loom's declarative and procedural components permits programmers to utilize the logic programming, production rule, and object-oriented programming paradigms within a single application.

For a detailed description of Loom see [MacGregor88].

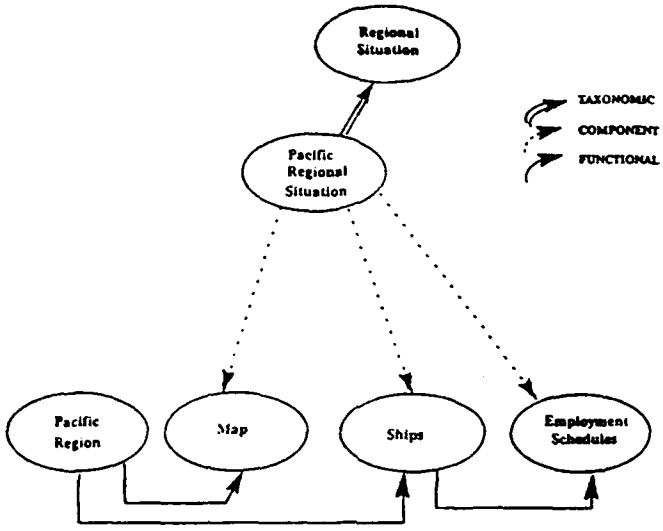

Figure 2: Fragment of Domain Model Containing *Pacific Regional Situation.*

4.3 The Modeling

Although SIMS contains a single model which includes descriptions of all relevant entities and relations among them in a uniform knowledge representation language, we conceptually divide SIMS' model into several parts:

- Application Domain Model
- Working Memory
- Service Capability Model
- Virtual Service Capability Model, and
- Goal Model

4.3.1 Application Domain Model

This contains a specification of objects in the domain of the SIMS system and their hierarchical organization into classes, as well as relations among classes. Taxonomic information is indicated by the standard *isa* relationship.

For example, the domain model for the Naval briefing application SIMS currently uses for demonstration purposes includes the concept *Pacific Regional Situation.* It is a subclass of *Regional Situation.* It stands in the *Component* relation to other objects which make up a situation briefing about a Pacific region: the name of the *Pacific Region* involved, a *Map* of that region, the *Ships* located in it, and the *Employment Schedules* of those ships. In addition, there are *Functional* relations between some pairs of these concepts, indicating that operators exist which can provide one object given another, as shown in Figure 2.

4.3.2 Working Memory

A collection of specific facts about the application domain which were obtained in the course of the current SIMS session. For example, a user inquiry about the location of a particular ship will result in a call to a particular database. In addition to being presented to the user, the information obtained from the database will be added to working memory, to avoid any need to retrieve it again in response to another request. At present, we do not address the possibility that such information may change in the course of a single SIMS session — other than by explicit application of operators, as described next.

4.3.3 Service Capability Model

Service capabilities are represented in SIMS as a collection of operators, each standing for a "primitive" operation provided by a service. The representation of each operator contains an indication of the *goal* it is capable of achieving, any *preconditions* that must be fulfilled for it to be applicable, a *method* for its application — i.e., a description of the actual call on the service and how to recast the results, if any — and *postconditions* of the operator's application. The postconditions typically involve modifications to working memory. Operator structure in SIMS is similar to that which is used in most bottom-up planners patterned after STRIPS.[Fikes71]

Below is an English rendition of an example, a simplified version of the operator *Employments-Given-Ship.*

Name	Employments-Given-Ship
Goal	Retrieve ship's employment schedule
Preconds	Ship is given
Method	Issue Retrieval command to Oracle database of quarterly employment schedules
Postconds	Assert knowledge of ship's employment schedule

4.3.4 Virtual Service Capability Model

Once SIMS' planner finds a plan — a sequence of operators capable of achieving the user's goal — it creates a new operator representing a *virtual service*, of a structure similar to that of the operators described above. The *method* of a virtual operator will be more complex, typically involving several service calls. This capability resembles the chunking of operators in SOAR [Laird87], and enhances the speed of SIMS work over time.

Virtual services that the SIMS designer assumes will be useful can be predefined.

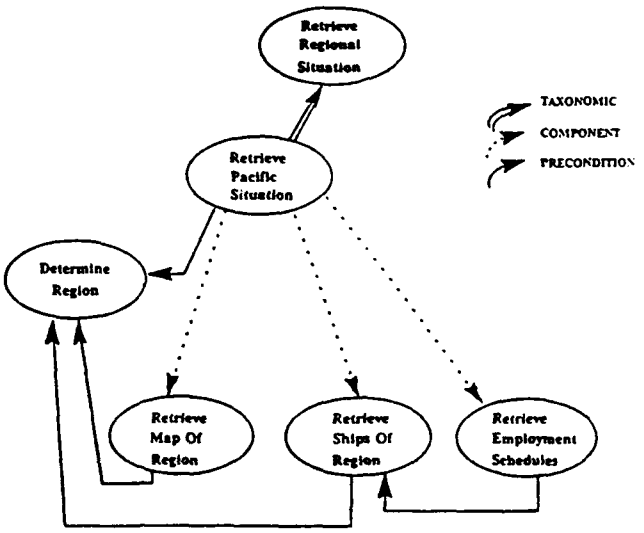

Figure 3: Fragment of Goal Model Containing *Retrieve Pacific Situation.*

4.3.5 Goal Model

User Goals are hierarchically organized: also, relations among them, and between them and domain objects, are explicitly represented. This is done, again, using the knowledge representation language Loom.

Loom provides great benefits to the modeler. The goal structure often is a reflection of the structure of the domain model. For example, the structure of *Pacific Regional Situation*, Figure 2, is inherited by the goals involved in retrieving the information concerning the situation in any specific region selected by the user. (See Figure 3). In Loom, one need only specify the goal *Retrieve Pacific Situation*, and the other goals are automatically generated by Loom's classifier mechanism.

The terminology of the goal model is currently also the service-independent language which is used by the user to present a task to the system. The need for the end-user to be familiar with Loom is an inconvenience which we intend to eliminate as we devise a more natural task description language in the future. However, even the current language is a major improvement over the need to be familiar with a multiplicity of access languages.

4.4 The Planning

The planning capabilities required to produce SIMS functionality is general enough to be implemented in a number of planning paradigms. The one selected will, however, have an effect on the performance of the system, and possibly on which problems may and may not be solvable. In the current version we have chosen to write a planner patterned generally after STRIPS.[Fikes71]

SIMS' planning process benefits from the fact that goals are represented in a language that allows their semantics to be expressed. In this respect it is quite different from other STRIPS-based planners. Every planner must match goals to operators which can be used to achieve them. In most cases such matching is done syntactically, using some form of pattern-matching. In SIMS, however, the matching is done using facilities provided by the knowledge representation language, which has access to semantic information not typically available in syntactic pattern matchers. This is of particular significance for goal reformulation, which we describe next.

4.5 Goal Reformulation

Every planner faces the problem of finding an operator that is capable of achieving a given goal. This is usually ascertained by comparing the desired goal state with a description of the operator's effects – part of the definition of the operator. The comparison is performed by some variant of the process of pattern matching. If the goal state does not match the patterns of any of the operators, an impasse is reached.

When SIMS' general planner fails a far more refined process of operator selection is possible. Both the goal state and the operators' effects are described in the model, represented in Loom, and the system thus has much detailed knowledge about them. In particular, Loom has a built-in reasoner, the *classifier*, that can be used to pre-compute and cache a large number of inferences about relationships among operators. If a given goal does not belong to the class of goals achievable by any operator, the planner engages in a knowledge-based process of *goal reformulation.* This process involves analyzing the knowledge base's representation of goals and relations among them in order to transform a goal that does not directly match any operator capabilities into a combination of goals that do.

Three classes of goal reformulation are addressed by SIMS. Goal repair, goal rephrasing, and goal modification. Examples of the first two classes have been implemented.

4.5.1 Goal Repair

Goal repair takes place when the user goal, as presented to SIMS, is not well formed, i.e., when it cannot be classified by Loom in the model. The repair component addresses this by attempting to replace elements of the goal until a well formed one is arrived at. This is done using knowledge of the domain and the structure of goals.

For example, a user may request that SIMS[3]

```
Display ships, Region = Sea of Japan
```

[3]This and all other examples in this paper are paraphrased in English, for readability.

The Loom classifier fails on this goal, since *region* is not defined as a role on ships. Navy personnel refer to the region within which a ship is employed by the term *emgeo*. *Region* happens to be a role on *map*, designating the geographic region it represents. Since *region* designates a geographic region, and *ship* does have a role (*emgeo*) designating a geographic region, SIMS replaces *region* with *emgeo*. This produces the well formed goal

```
Display ships, Emgeo = Sea of Japan
```

4.5.2 Goal Rephrasing

Goal rephrasing is engaged in when a goal is well-formed, but no plan is associated with the goal. The process of rephrasing involves the use of knowledge of the structure of the goal hierarchy to replace the goal with a semantically identical combination of others.

Several strategies for this have been identified.

Conjunction: Some unachievable goal state state may be the conjunction of several other goals. In this case, the original goal is reformulated as the sequence of the conjunct.

For example, the goal class **Retrieve Ship Status** is not satisfiable by any operator. However, this class is represented in the model as the union of **Retrieve Ship Location**, **Retrieve Ship Course**, and **Retrieve Ship Employment**. The original goal is reformulated as the conjunction of the others. Note that the relationship among these goals is inherited from domain model information similar to that described in Figure 2.

Specification: Some unachievable goal may be a member of a class which is the union of several other goal classes. In this case, the original goal is reformulated as the goal of attempting to achieve each of the subordinate goals until some attempt is successful.

For example, assume that the location of ships are stored in databases, separately for each country of origin. The class **Retrieve Ship Location** is thus the union of **Retrieve Ship Location for Ship of Country X**, for each country **X**. Given a ship of unknown origin, the goal of determining its location can be reformulated as the goal of attempting to find its location in each of the various countries' databases, until success is achieved.

Subsumption: Some unachievable goal may be subsumed by an achievable one. The former is reformulated as the latter.

For example, assume that ship displacement is associated not with individual ships, but rather with ship classes. Thus the goal **Retrieve Ship Displacement** will be subsumed by **Retrieve Displacement of Ship Class** — in fact, this relationship will be inherited from the relationship between the domain objects **Ship** and **Ship Class**. The goal of determining a ship's displacement will be reformulated as that of determining the displacement of ships in the given ship's class.

4.5.3 Goal Modification

Goal modification is engaged in after the failure of goal rephrasing to obtain a goal for which a plan can be found. This process attempts to replace the original goal with one which would result in achieving the same *purpose*. Determination of the purpose of the original goal is not simple, and requires establishing the relationship between that goal and higher level goals the user may have. It requires knowledge of the overall structure of the user-SIMS interaction.

To illustrate what is involved in goal modification, let us examine the following scenario. The user is involved in the planning of a search and rescue mission for a downed plane. The user has chosen to send a rescue ship, and posts the immediate goal

```
Send ship A
```

If the system fails to execute this goal, due to the fact that ship A has a powerplant problem, it would be reasonable to replace it with the goal

```
Send ship B
```

for another, currently available, ship B.

However, one cannot conclude that these two goals are always equivalent, and that the first can always be modified into the second. If the user is planning for a scheduled crew rotation, and for that purpose posts the goal of sending ship A to port, it would be unacceptable to substitute for that the goal of sending ship B.

4.6 The SIMS Prototype System

ISI has completed implementation of a SIMS prototype system in the domain of daily Navy briefings at CINCPACFLT. This demonstration system permits access to four separate database tables in an Oracle database containing information from this domain, in a manner transparent to the user.

To use the system, a request for data is stated as a goal in the LOOM knowledge representation language. Problematic requests may be identified and reformulated by the system: Two types of reformulation are currently implemented. "Goal Repair" is used to replace erroneous terms used in a user's goal until a well-formed goal is arrived at. Modeled knowledge of semantic similarity between the terms is used as a guide. "Goal Rephrasing" is used to replace a goal with which no operator is associated, with another, or a combination of others, for which plans are available. Modeled knowledge of the structure of possible user goals and domain concepts is used for this process.

Once planning is successfully completed and a sequence of operators for achieving the user's goal is found, SQL queries are automatically generated, and the appropriate databases are accessed remotely over a LAN. The returned data is filtered and combined to satisfy the user's original query.

4.7 Implementation of the SIMS Prototype

The main objective of SIMS is to provide access to a multiple, heterogeneous database network using a single query language. Because instances of concepts are reified in DBs, we could not use the reification mechanism provided by LOOM through the ABOX. The reason being that by definition the ABOX does not provide for multiple reification of an instances (in a sense, similar instances are "instances" of that one instance), and there are no mechanism to access databases. Reification is accomplished by providing explicit links between conceptual objects/relations and their reified terms in the DBs and a set DB access functions and a planning component to plan DB access necessary because of the added complexity of multiple reifications (their equivalency as well as different access paths).

4.7.1 Models

SIMS has 3 different kinds of models;

- the Application Domain Model (APM)
- models of the different databases (DBM).
- the Database Terminological Model (DTM)

APM The APM is a model of the domain the user is concerned with. It consist of the conceptual objects found in the domain and the relations relating those objects.

DBM The DBM is a high level model of the databases (DB) actually used by the application to maintain their data. This declares the important characteristics of a particular class of DB, which allows us to automatically generate a parser to process instances of the database. DBs are defined in the following manner;

```
(def-sims-db <type>
        :host <pathname-specification-type>
        :access-mechanism <access-mechanisms>
        &optional :key <index-type>
                  :table <table-specification> ...)
```

<type> is the type/class of DB to be defined

:host host name of the DB

:access-mechanism one of :RELATIONAL, :INDEXED, :QUERY-INDEXED

:key <index-type> if specified implies that operations on this DB requires a key/index specified by <index-type>

:table <table-spec> Implies that the database is grouped into tables of relations

:relations <relation-tuple> if specified implies that relations in this DB is not further organized into tables.

E.g.,

```
(def-sims-db LCA
        :access-mechanism :QUERY-INDEXED
        :host PATHNAME
        :key RELATION
        :query-key STRING
        :table SYMBOL)
```

This means that LCA DBs have the following characteristics;

- it uses the QUERY-INDEXED access mechanism, ie., that operations on this type of DB requires both a key and the query type;
- the relations in this DB is grouped into tables;
- the key required is an existing relation;
- the query key required is a string.

In addition to the modelling of conceptual properties of a DB, a set of DB operations is required. This module contains functions (LOOM methods) that allow SIMS to interface to the actual DBs — it is a functional specification of what DB operations are supposed to do with a standard set of parameters. This makes it easier for the application builder to add servers to the system, since it is known exactly what kind of functionality must be provided/declared to SIMS. Currently we've implemented the following interface functions;

- DB open & closing functions. For simple DB such as flat file, it may be provided by the language itself (OPEN), for more complicated DBs, eg., ORACLE, it requires the application builder to not only establish a connection (usually via TCP), but also functions to prompt for account and password (or even mounting of specific databases).
- Query reformulating functions. Functions to change the SIMS query into the query language of the DB.
- DB accessing function. Functions to actually query the DB, we've defined only "atomic" queries at the moment, ie., queries specifying a single goal and a constraint for data generation queries and two constraints (corresponding to the domain and range of a relation) for a filtering query. (later we will add side-effecting DB operations, eg., updating of entries, deletion and data creation (also tables?)). Since the type of answers returned by a DB query is not necessarily

what is required, eq., a DB may return the entire record while only a particular field of the record is of interest, these functions need to be able to perform this kind of extraction too.
[Set of functions to deal with problems, eg., how a problem in opening a DB is handled, whether non-existence of a data (as opposed to one that cannot satisfy a constraint) is reported as error or ignored, etc.
Hooks for more intelligent accessing functions that can contain multiple constraints and goals so that a query may possibly be answered with a single DB access rather than multiple accesses.]

DTM The DTM is a reification model, ie., it denotes how application domain concepts/relations are reified in actual DBs, how the data is actually represented. This is the interface between the conceptual level and the "real world" of data processing. The DTM consist of the terms actually used in the databases that is used to store data in the application and what domain concepts/relations it correspond to in the APM. Since a particular APM concept/relation may appear in more than one DB, this implies that the mapping of APM → DTM is 1-many. Data representation at the DB level is defined at the level of numbers, strings and other primitive data types.

Each APM concept that is reified as DB instances has a relation DATA-ACCESSORS containing the set of reified DB instances which are represented by the concept DB-ACCESSOR;

```
(defconcept db-accessor
  :is (:and :primitive
       (:the table symbol)
       (:the db database)
       (:the db-term concept)))
```

To eliminate the possibility of DB terminological terms superceding or redefining domain terms or terms from another database (a relation may be used to denote different things in different DBs), all relations of a DB are defined in a separate knowledgebase partitions (the conceptual analog to packages in Common Lisp).

E.g.,

```
(def-sims-dbtable nsn
        :type :lca
        :key national_stock_number
        :query-key "NSN"
        :relations
        ((national_stock_number (alphanumeric 13) transcom-cargo)
          (geographic_area_code (alphanumeric 1) geographic-location)
```

```
(document_number (alphanumeric 14) transcom-requisition-order)
(unit_of_issue (alphanumeric 2) cargo-units)
(quantity (alphanumeric 5) cargo-quantity)
(ship_date (alphanumeric 8) departure-date)
(port_of_embarkation (alphanumeric 3) embarkation-location)
(port_of_embarkation_receipt_date (alphanumeric 8) delivery-date)
(port_of_embarkation_lift_date (alphanumeric 8) lift-date)
(voyage/flight_number (alphanumeric 7) passage-number)
(project_code (alphanumeric 3) project)
(issue_priority_designator (alphanumeric 1) priority)
(required_delivery_date (alphanumeric 8) scheduled-arrival-date)
(tcn (alphanumeric 16) transcom-shipment)))
```

The 3-tuple for the relations are the DTM term, eg.,

```
(national_stock_number (alphanumeric 13) transcom-cargo)
```

The first tuple is the term actually used as the DB relation, the second tuple is the data type specification, in this case an alphanumeric string of length 13. The third tuple is the APM concept of this term.

Note that the data type specification is often much broader than is actually the case, eg., in the above definition, many of the terms are actually represented by numbers, eg., quantity, unit_of_issue, etc. or have very specific format, eg., dates in yy/mm/dd format.

[In addition to the above models, there is also a small KB in SIMS consisting of concepts and relations useful in defining the above models, eg., a small "upper model" of more abstract concepts useful for domain modelling and primitive data types (eg., numbers, strings, number ranges, etc.)]

4.7.2 Planner

SIMS performs two basic kind of planning;

- Actual DB accessing plans that may access multiple DBs.
- Plans to satisfy a complicated query consisting of multiple plans of the above nature.

The planner produces 3 different kinds of operators. At the lowest level, the data retrieval operators provides a direct interface to the databases and work only with specific given arguments, and there are one operator for each given argument type that access a particular set of databases. Above that is the "meta" operators, one for each goal, they generate data retrieval operators to handle data retrievals for specific argument types. The last type of operators are the "complex" operators that is used to satisfy complicated queries — queries composed of multiple data retrievals.

DB accessing plans These are operators that actually access the DBs to retrieve data (retrieval operators) or filter predicates to ensure that a concept with a particular relation have a specific value (filter operators). This task complicated by the fact that:

- the data may appear in a form different from the one desired by the user.
- the arguments passed may appear in a form different from the one required for accessing the DB where the data is stored.

If any of the above happens, then it will be necessary to transform data from one form to another, this will probably require accessing multiple DBs. Data transformation relies on the following;

- data is transformed to another form that is also a reification of the same APM concept as the original concept. For example, given SHIPNAME, "JFK", a reification of the APM concept SHIP, transform it to a UIC (unit identification code), another reification of SHIP.
- At least one DB contains two reification of the APM concept of the data in question. Furthermore, it must be possible to trace a path starting from the original form, from one DB containing that form and another reification, not necessarily the desired reification, in a chain until a DB containing the desired reification and another reification is found. Ie.,
  Given Dg = given data form, Dd = desired data form
  (All Ds are different reification of the same APM concept/relation)
  find path, in-DB(DB1, Dg, D1)...in-DB(DBn, Dn, Dd)
  then, in order to transform Dg to Dd, we first perform in order,
  transform Dg to D1 in the database DB1 and so on, finally
  transform Dn to Dd in database DBn

This is implemented as follows, given a goal to retrieve, GOAL, and the argument, ARG;

- find out the APM class of ARG, this is done by checking the domain or range of GOAL.
- find out the APM reification of ARG, Dd, check which reification of the APM concept match it (a type check of DB-ACCESSORs in the DATA-ACCESSOR relation of the concept).
- check if operators have been defined for this goal, by checking if the relation DB-FILTER-OPERATOR or DB-RETRIEVAL-OPERATOR has been defined, if not, define the Meta-DB operator,

This operator is generalized over all reifications of the APM concept, not just for a specific argument type. One exist for each APM concept/relation that is encountered, this operator does not access DBs but generate DB operators for specific reifications of the APM concept-relation. This has the following form;

```
(defmethod <name> (?given)
   :situation (ARG-OF-TYPE <APM of ?given> ?given)
   :action <call planner>>)
```

ARG-OF-TYPE is a predicate that takes a APM concept/relation and a DTM concept instance and returns true iff that DTM concept is a reification of the APM concept/relation.

[<name> is the goal, APM concept/relation, by convention, a data retrieval operator will be called GET-<APM-concept-name>-INSTS and filter operators are called FILTER-<APM-concept-name>-INSTS]

The mapping relation between a Meta-Db operator → Db operator is 1-many, but we do not generate the operators all at once but on a as-needed basis. SIMS keeps tracks of access path that has been used to generate DB operators when given specific arguments so that it does not need to duplicate computation already done. The DB access path planning is done as follows;

- DBs that contain both GOAL and GIVEN are found, and ordered as follows;

  ```
  goal-type    given-type    no conversion
  goal-type    given-term    given-term needs to be converted
  goal-term    given-type    goal-term needs to be converted
  goal-term    given-term    both terms need to be converted.
  ```

  This ordering ensures that when a suitable path is found, the path is optimal, ie., for example, if the first type is possible, it is best because the DB contains the relations that corresponds exactly to what is given and what is desired, while the rest requires some conversion.
- find DBs that contains GOAL,
  For each DB that contains GOAL
  if it also contain Dd then done
  if it contains Di where Di $\neq$ Dd, then recurse using Di and Dd
- If a path is found, define an operator using the data-path, save DBs not yet used in the above step to be used if alternative operators are needed. [see section on operators to see how they are used and defined]

These are the operators that perform the actual DB accessing, it is generated by the Meta-DB operators and each operator works for a specific APM reifications only, ie., as many DB operator are generated as there are reifications of an APM concept/relation (we generate them on a as-needed basis though).

These operators have the following form;

```
(defmethod <name> (?given)
   :situation (<db-term> ?given)
   :action <Db accessing code>)
```

e.g.,

```
(DEFMETHOD GET-EMPLOYMENT-AREA-INSTS (?SHIPNAME)
  :SITUATION (SHIPNAME ?SHIPNAME)
  :TITLE "(ACC3663 ACC3659 ACC3648 ACC3647)"
  :ACTION ((WHEN (SETQ ?SHIPNAME
                       (CONVERT-ARGUMENT '(|I|ACC3648(uic uchar)
                                           |I|ACC3647(shipname uchar))
                                         ?SHIPNAME))
            (PERFORM (GENERATE-DB-INSTANCES |I|ACC3659(uic empskd2)
                                            |I|ACC3663(emgeo empskd2)
                                            ?SHIPNAME)))))
```

This operator is used to find the EMPLOYMENT-AREA given a SHIPNAME. But the DB that contains both of the reified terms, accept a UIC instead of SHIPNAME. The given argument must be converted to a UIC first, done through the UCHAR table of database, only then can the retrieval proceed. Hence a "simple" retrieval here actually involves using two separate DB tables.

Composite Plans These are constructed when the query is composed of more than one subquery. This differs from data retrieval operators in that it does not contain code to access the DBs but call operators that do. It triggers data retrieval operator construction and perform the appropriate variable bindings to ensure only instances satisfying the query is returned. This is necessary especially in cases using the filter operators for a set of instances, since in general DBs do not allow one to pose a set query, eg.,

```
Q = (:and (employment-area "JFK" ?region)
          (employment-area ?ships ?region)
          (class ?ship "AGGRESSIVE")
          (readiness-level ?ships 1))
```

[Note that queries always return sets as answers] In this query, the second subquery will in general return a set of SHIPs, so that the third subquery is a predicate to filter out SHIPs that do not have READINESS-LEVEL = 1. One cannot pose just one query that uses the entire set of SHIPs, but must instead pose it one at a time, for each SHIP — hence the operator itself must keep track of instances that work.

In effect, the query, when generalized over the argument type from DTM to its APM concept, provides the abstract specification of retrieval and conditions that needs to be satisfied. Argument differences (from the original differences) are resolved at the lower level by the meta-DB operator, ie., if none of the existing DB operators can be used, then the meta-DB operator will try to generate a new operator to take care of this case.

For the above query, the following operator is generated; E.g.,

```
(DEFMETHOD ABLE-SHIPS (???SHIP ???SHIP-CLASSES ???READINESS-LEVEL)
  :SITUATION (:AND (ARGUMENT-OF-TYPE SHIP ???SHIP)
                   (ARGUMENT-OF-TYPE SHIP-CLASSES ???SHIP-CLASSES)
                   (ARGUMENT-OF-TYPE READINESS-LEVEL ???READINESS-LEVEL))
  :ACTION
  ((LET (?S ?R)
     (SETQ ?R (PERFORM (SIMS-RETRIEVE |R|EMPLOYMENT-AREA ???SHIP)))
     (SETQ ?S (PERFORM (SIMS-SET-RETRIEVE |C|SHIP ?R)))
     (SETQ ?S (PERFORM (SIMS-SET-FILTER |R|CLASS ?S ???SHIP-CLASSES)))
     (SETQ ?S (PERFORM (SIMS-SET-FILTER |R|OVERALL-READINESS
                                        ?S ???READINESS-LEVEL)))
     ?S)))
```

Note that an additional argument to specify the goal of this query is optionally supplied, in this case ABLE-SHIPS. Only one method is generated for each query and it is usable for all reifications of the required arguments, ie., this method will work even if we used say, the UIC of the ship instead of the SHIPNAME and no replanning is required. Note that some of the DB accessors are set operations.

In addition, since the goal is now defined as a concept with the givens as relations on it, it is now possible to to ask for instances satisfying the above query simply by asking for instances of this concept, ie., for the above example, we can now ask for

```
(ABLE-SHIPS "N23245" "SUPPLY" 2)
```

i.e., instances of ships in the same region as "N23245" whose class is "SUPPLY" and whose readiness-level is 2.

4.8 List of Publications

- E. Hovy and Y. Arens. Automatic Generation of Formatted Text. In *AAAI-91: The Tenth National Conference on Artificial Intelligence.* Anaheim, CA, July 14–19, 1991.
- Y. Arens, E. Hovy and M. Vossers. Categorizing the Knowledge Used in Multimedia Presentations. In *AAAI-91 Workshop on intelligent Multimedia Interfaces.* Anaheim, CA, July 14–19, 1991.
- Y. Arens, L. Miller, and N. K. Sondheimer. Presentation Design Using an Integrated Knowledge Base. In *Intelligent User Interfaces.* Edited by Joseph W. Sullivan and Sherman W. Tyler. Addison-Wesley, 1991.

- Y. Arens, E. Hovy and M. Vossers. Organizing the Knowledge Needed for Multimedia Communication. (Presentation to panel — Multimedia in AI: Challenges and Opportunities.) In *CAIA-91: Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications; Volume II: Visuals.* Miami Beach, FL, February 24–28, 1991.
- Y. Arens and E. Hovy. Text Layout as a Problem of Modality Selection. In *KBSA-5: Proceedings of the 5th Annual Knowledge-Based Software Assistant Conference.* Syracuse, New York, September 24–28, 1990.
- Y. Arens and E. Hovy. How to Describe What? Towards a Theory of Modality Utilization. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society.* MIT, Cambridge, Massachusetts, July 25–28, 1990.
- Y. Arens. Services and Information Management for Decision Support. *AISIG-90: Proceedings of the Annual AI Systems in Government Conference.* George Washington University, Washington, DC. May 7–11, 1990.
- Y. Arens. A Knowledge-Based Multi-Modal Interface. *AISIG-90: Proceedings of the Annual AI Systems in Government Conference.* George Washington University, Washington, DC. May 7–11, 1990.
- E. Hovy and Y. Arens. Allocating Modalities In Multimedia Communication. *Working Notes: AAAI Spring Symposium on Knowledge-Based Human-Computer Communication.* Stanford University, California. March 27–29, 1990.
- J. Pavlin and R. L. Bates. SIMS: Single Interface to Multiple Systems. Invited Paper at *Tenth International Computer Symposium.* University of Dubrovnik, Yugoslavia, 1988. (Also available as ISI Research Report ISI/RR-88-200.)
- J. Pavlin and R. L. Bates, SIMS: A Uniform Environment for Planning and Performing Users' Tasks. *Proceedings of First International Conference on Industrial and Engineering Applications of AI and Expert Systems.* Tullahoma, TN, 1988.

5 Important Findings and Conclusions

SIMS has made progress along all significant dimensions of its approach to the software integration problem:

- SIMS operates using a general purpose planner, for which individual queries to databases serve as operators.
- All data services available through SIMS are accessed uniformly by stating requests in the LOOM knowledge representation language. Requests are analyzed using both built-in LOOM reasoning facilities and SIMS-specific ones.

- SIMS has developed models of four different database tables containing actual (sanitized) information from the CINCPACFLT domain. The data in these tables is accessed and combined without and modification to the DBMS itself.
- Several goal reformulation strategies have been developed and implemented. In some cases, SIMS can correct syntactically flawed user requests. It is capable of replacing goals which it cannot achieve with equivalent combinations of goals which it can.

6 Significant Hardware Development

None.

7 Special Comments

Research on this topic has been continuing at ISI, under the *Services and Information Management for decision Systems* project. Using the additional familiarity with the problem gained in the course of the work reported above, coupled with the better understanding of the issues acquired in the course of that work, the SIMS demonstration system has been reimplemented along different — although fundamentally similar — principles.

8 Implications for Future Research

The SIMS approach provides important benefits to the multiple-service system user. Among them are:

- Relieving the user of the need to be familiar with available services;
- Reducing the cost and the extent of the revision required to add a new service to the system, or to modify an existing one;
- Ensuring that a newly added service is utilized in the execution of all user requests for which it may be appropriate.

The goal reformulation mechanism in SIMS, by using a declarative model of goals and their interrelationships, reduces the chance of planner failure due to the inflexibility of matching techniques. Greater planner applicability provides for better responsiveness and adaptability.

In addition, the rigorous structure imposed on goals and operators, and the sophisticated knowledge representation language used, supports faster turn-around in large-scale operations. These software engineering gains will result from easier building and maintainability of the composite system.

References

[Erman87] Erman, L. D., J. S. Lark and F. Hayes-Roth, "ABE: An Environment for Engineering Intelligent Systems," Technical Report TTR-ISE-87-106, Teknowledge Inc., Palo Alto, CA, 1987.

[Fikes71] Fikes, R. and N. J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2, 1971, pp. 189-208.

[Laird87] Laird, J. E., A. Newell and P. S. Rosenbloom, "SOAR: An Architecture for General Intelligence," *Art. Int.*, Vol. 33, No. 1, 1987.

[MacGregor88] MacGregor, R., "A Deductive Pattern Matcher," *Proceedings of AAAI-88, The National Conference on Artificial Intelligence*, St. Paul, MN, August, 1988.

[Neches85] Neches, R., W. R. Swartout and J. D. Moore, "Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development," *IEEE Transactions on Software Engineering*, Vol. SE-11, No. 11, November 1985, pp. 1337-1351.

[Pavlin88a] Pavlin, Jasmina and Raymond L. Bates, "SIMS: Single Interface to Multiple Systems," Invited Paper at *Tenth International Computer Symposium*, University of Dubrovnik, Yugoslavia, 1988. (Also available as ISI Research Report ISI/RR-88-200.)

[Pavlin88b] Pavlin, Jasmina and Raymond L. Bates, "SIMS: A Uniform Environment for Planning and Performing User's Tasks," *Proceedings of First Int. Conf. on Industrial and Engineering Applications of AI and Expert Systems*, Tullahoma, TN, 1988.

[Swartout88] Swartout, William, Ed. "DARPA Santa Cruz Workshop on Planning," *AI Magazine*, Vol. 9, No. 2, 1988, pp. 115-130.

9 DD Form 1473

REPORT DOCUMENTATION PAGE

UNCLASSIFIED

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILTY OF REPORT APPROVED FOR PUBLIC RELEASE DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(s)	5. MONITORING ORGANIZATION REPORT NUMBER(S) Information Officer (S&IO/TIO) Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203-1714 (703) 696-2452
6a. NAME OF PERFORMING ORGANIZATION: USC/Information Sciences Institute 6b. OFFICE SYMBOL	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) 4676 Admiralty Way Marina del Rey, CA 90292	7b. ADDRESS (City, State, and Zip Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION [redacted] ARPA 8b. OFFICE SYMBOL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA-903-87-C-0641
8c. ADDRESS (City, State, and ZIP Code) Universtiy of Southern Calif at San Diego (A-034) Scripps Institute of Oceanography 8603 LaJolla Shores Drive San Diego, CA 92093-0234	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (INCLUDE SECURITY CLASSIFICATION)

SIMS

12. PERSONAL AUTHOR(S)

Yigal Arens

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF EPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	92 08 31	22

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software Integration, Multidatabases, Database Retrieval, Planning

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Services and Information Management for decision Systems (SIMS) project integrates several computer services into a single, seamless system, as seen from the user's perspective. This relieves the user of the need to be familiar with all the services available from the system, enabling the user to make requests as though he or she were dealing with a single entity. SIMS analyzes the user request, plans the necessary sequence of calls on the services and executes them. Until now, the only approach to the problem of software integration has been to build custom systems that support an integrated view of the underlying servers. Such systems are expensive, are suitable only for the application for which they were carfted, are not very flexible with respect to functionality, and are difficult to modify. In contrast, SIMS would allow maintainers to asimilate servers into the system through a much less painful process of building a declarative model of the servers, and allowing the SIMS planner to perform the actual integration task.

continued on back

20. DISTRIBUTION/AVAILBILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	U/U
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

UNCLASSIFIED